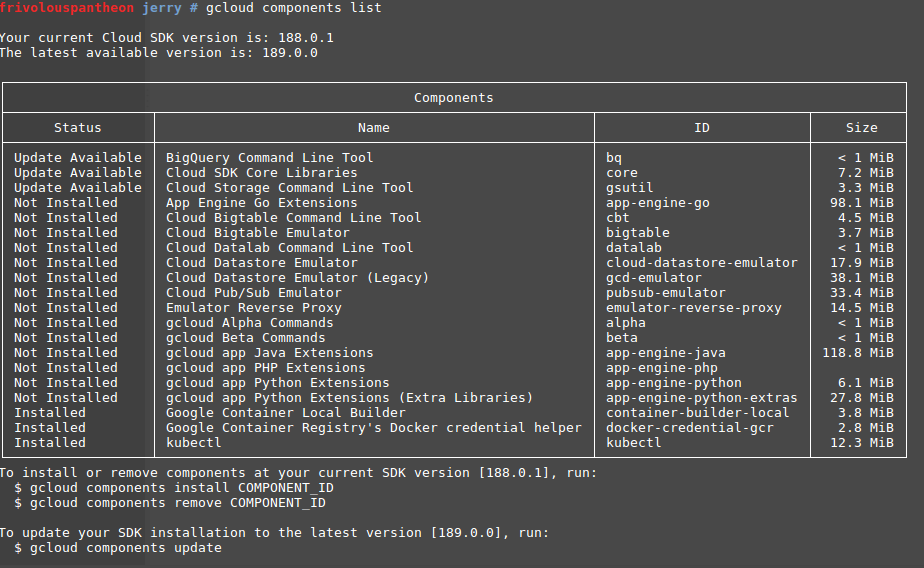
* Create Google Cloud Account
* Install gcloud SDK
  + <https://cloud.google.com/sdk/>
* After installing the
  + Minicube – Single node kubernetes
    - For installation - <https://www.youtube.com/watch?v=_vHTaIJm9uY>
  + Kubernetes multi-node cluster in Google Cloud Platform

Commands to execute

1. gcloud components list



If kubectl is not listed as installed then install it using the command

gcloud component install kubectl

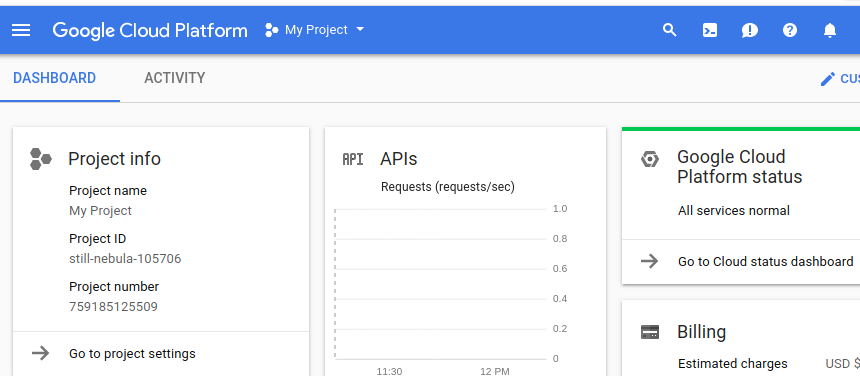
Ref:<https://cloud.google.com/sdk/gcloud/reference/components/>

* Login to using the CLI:

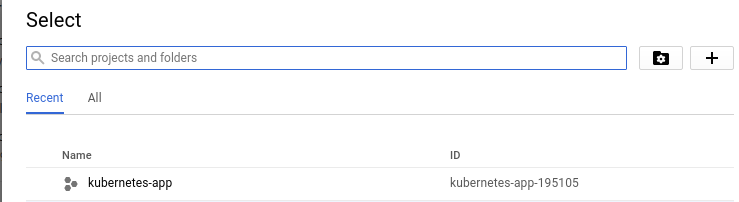
gcloud auth login

Enter your username and password

Go to your gcloud console and create a project



Next, Click + sign And assign your project name



Next in the CLI

gcloud config set project [PROJECT\_ID]

gcloud config set compute/zone [COMPUTE\_ZONE]

OR

gcloud init

Enter option 1.

And login with your username

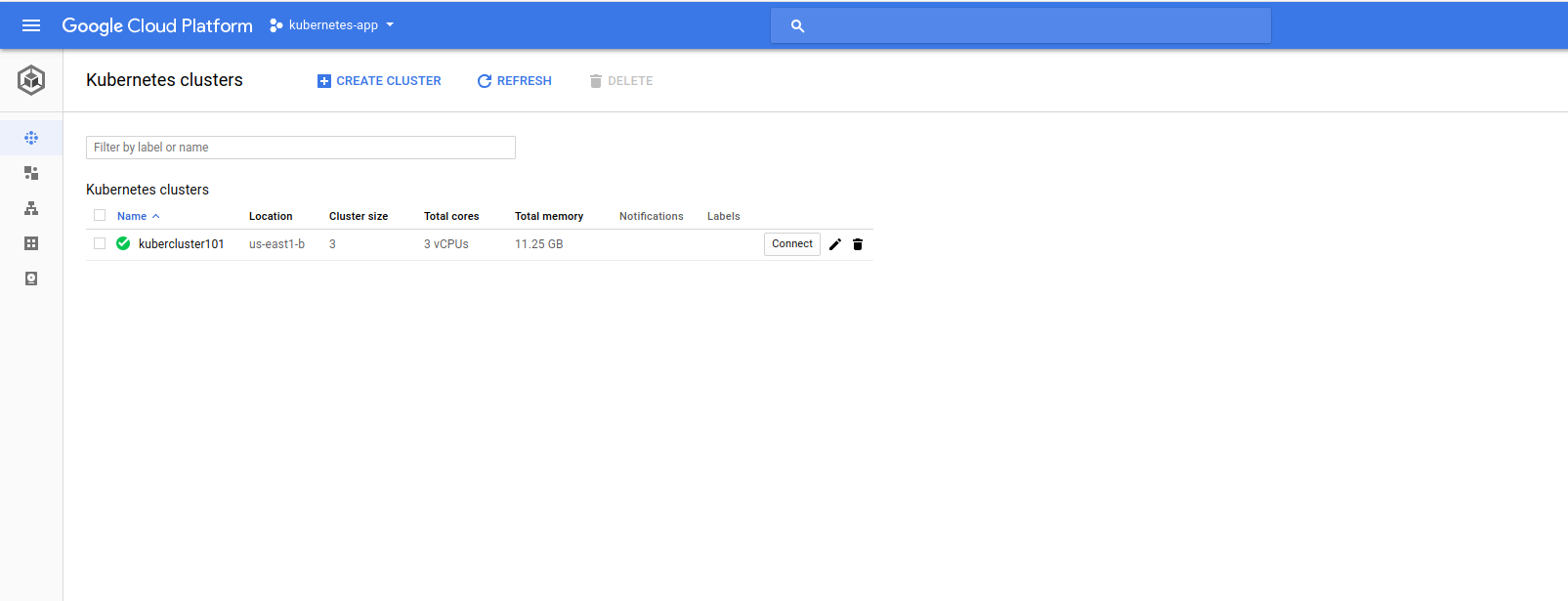
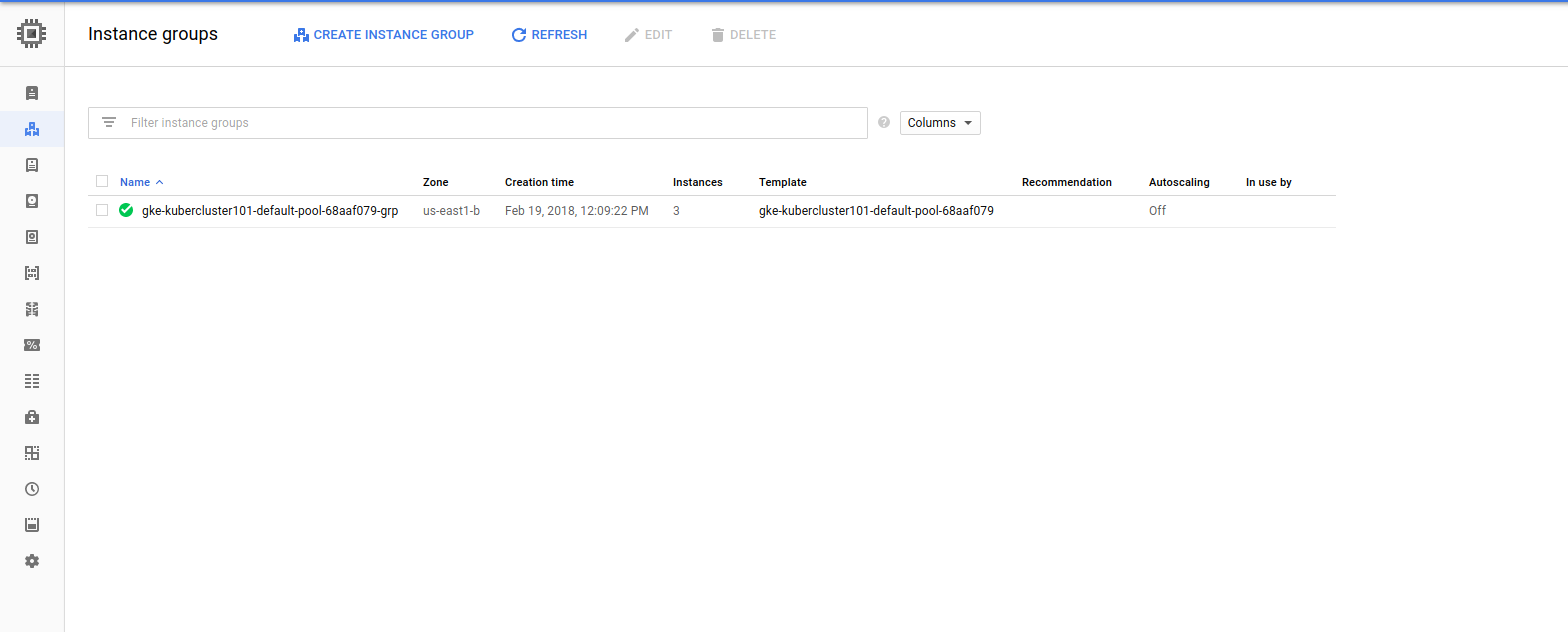
Select the project you just created.

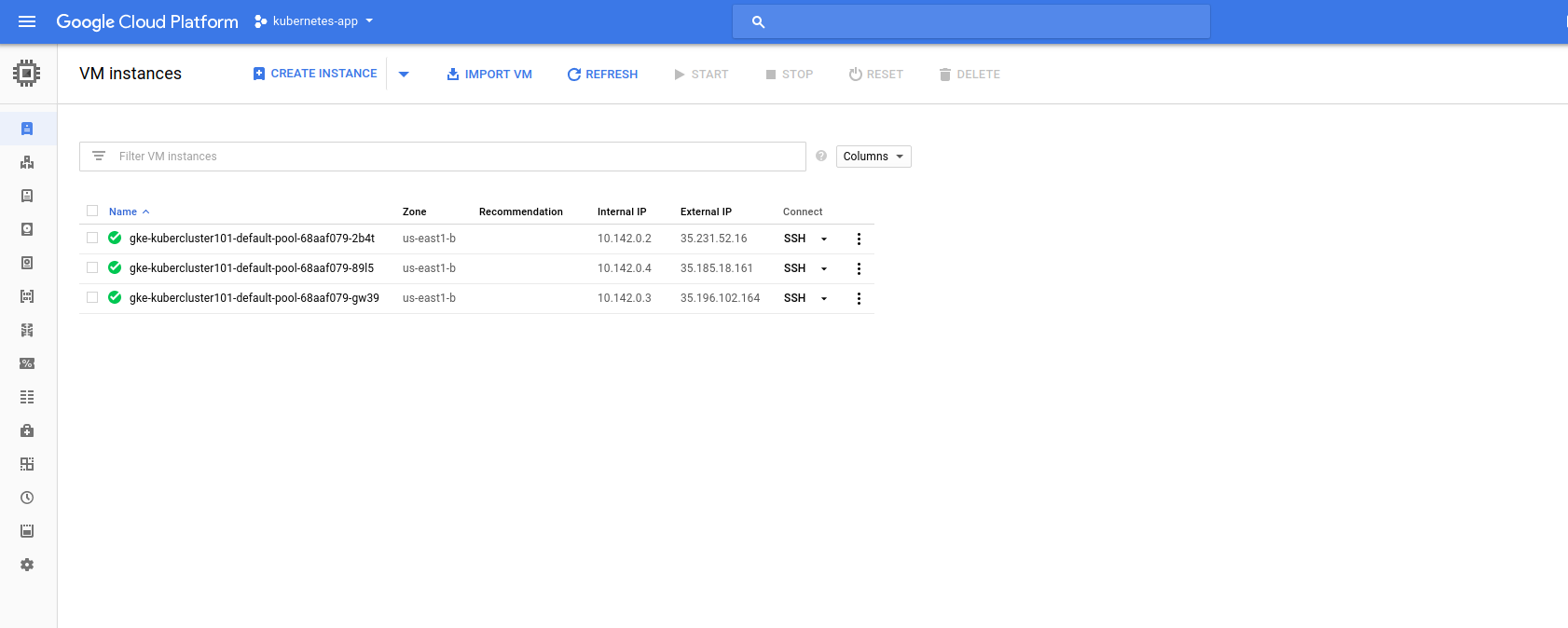
Select the configure zone

2. Create Kubernetes Cluster.

A [cluster](https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-architecture) consists of at least one cluster master machine and multiple worker machines called nodes. Nodes are [Compute Engine virtual machine (VM) instances](https://cloud.google.com/compute/docs/instances/) that run the Kubernetes processes necessary to make them part of the cluster. You deploy applications to clusters, and the applications run on the nodes.

Commands : gcloud container clusters create kubercluster101





Ref: <https://kubernetes.io/docs/setup/pick-right-solution/#hosted-solutions>

3. Push Docker images to Container Registry

Create a Docker file

# Import Python runtime and set up working directory

FROM python:2.7-slim

WORKDIR /app

ADD . /app

# Install any necessary dependencies

RUN pip install -r requirements.txt

# Open port 80 for serving the webpage

EXPOSE 80

# Run app.py when the container launches

CMD ["python", "app.py"]

Create requirements.txt

Flask

Create app.py

from flask import Flask

import os

import socket

app = Flask(\_\_name\_\_)

@app.route("/")

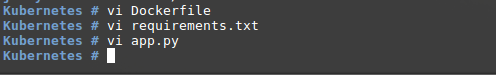
def hello():

    html = "<h3>Hello, World!</h3>"

    return html

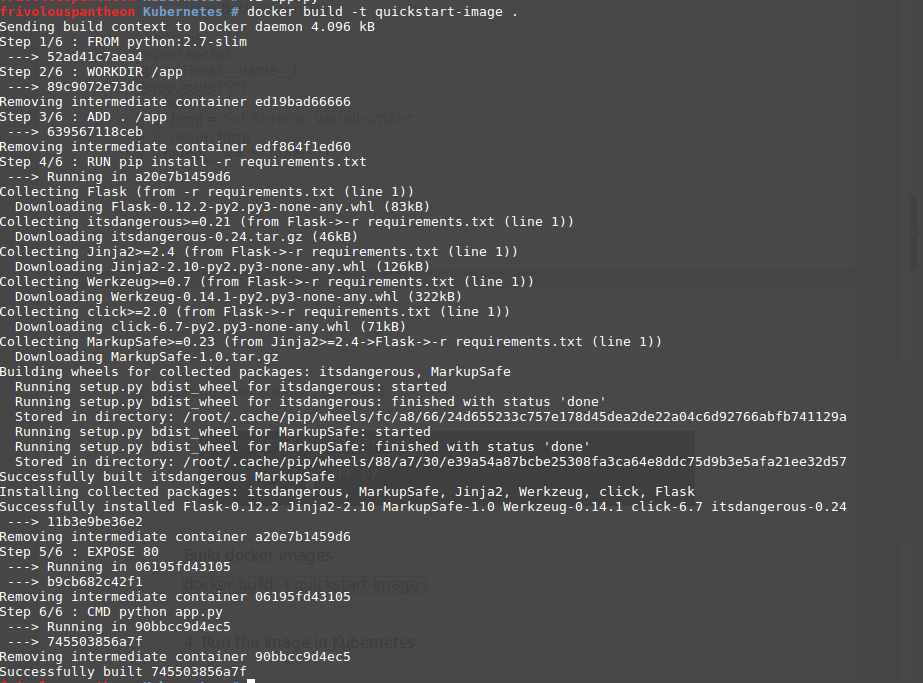
if \_\_name\_\_ == "\_\_main\_\_":

  app.run(host='0.0.0.0', port=80)



Build docker images

Command : docker build -t quickstart-image .



## **Tagging your image**

Before you push your Docker image, you need to tag it with its registry name. Tagging your Docker image with a registry name configures the **docker push** command to push the image to a specific location.

The registry name format is below:

[HOSTNAME]/[PROJECT-ID]/[IMAGE]

where

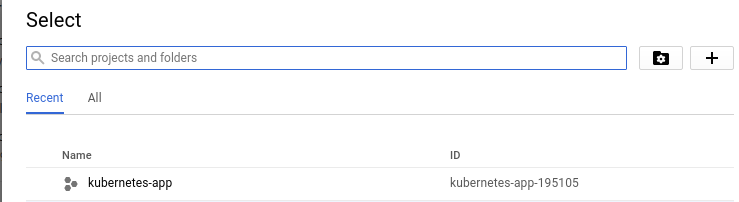
* **[HOSTNAME]**is the gcr.io hostname
* **[PROJECT-ID]** is your Google Cloud Platform Console [project ID](https://support.google.com/cloud/answer/6158840)
* **[IMAGE]** is your image's name

To tag your Docker image for Container Registry, run the following command:

docker tag [IMAGE] [HOSTNAME]/[PROJECT-ID]/[IMAGE]

For example:

docker tag quickstart-image gcr.io/my-project/quickstart-image



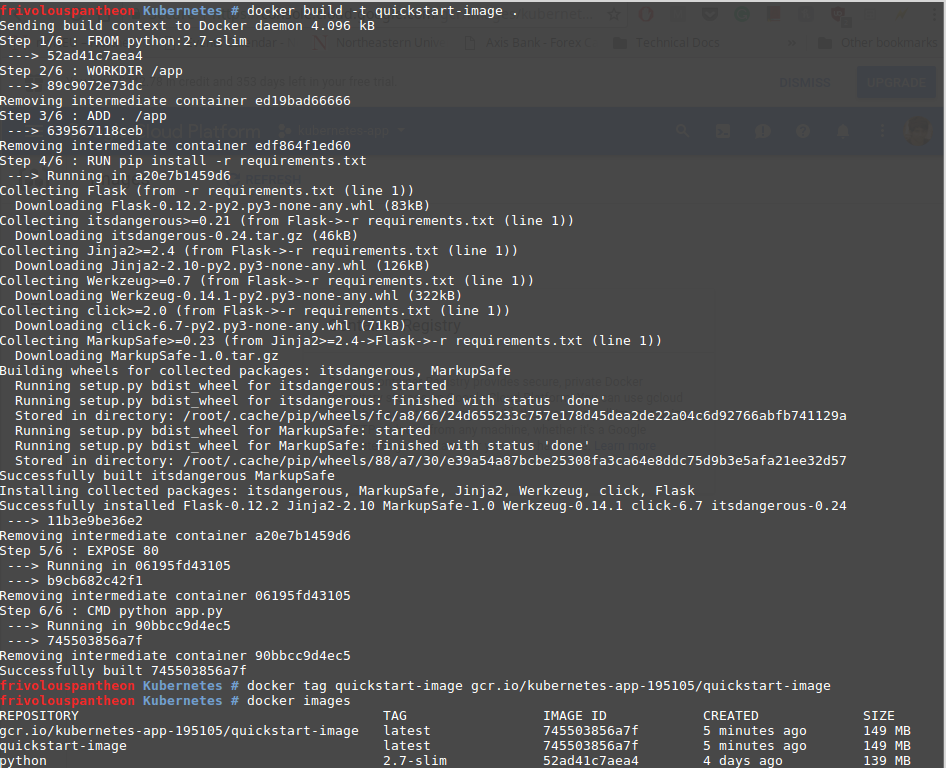
## Pushing your image

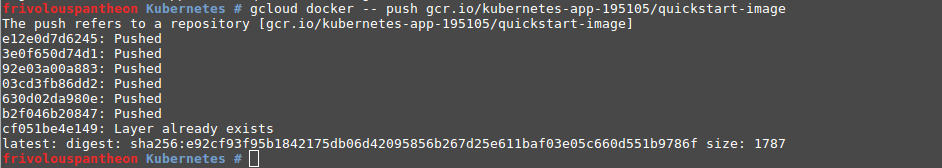
To push your Docker image to Container Registry, run the following command:

gcloud docker -- push [HOSTNAME]/[PROJECT-ID]/[IMAGE]

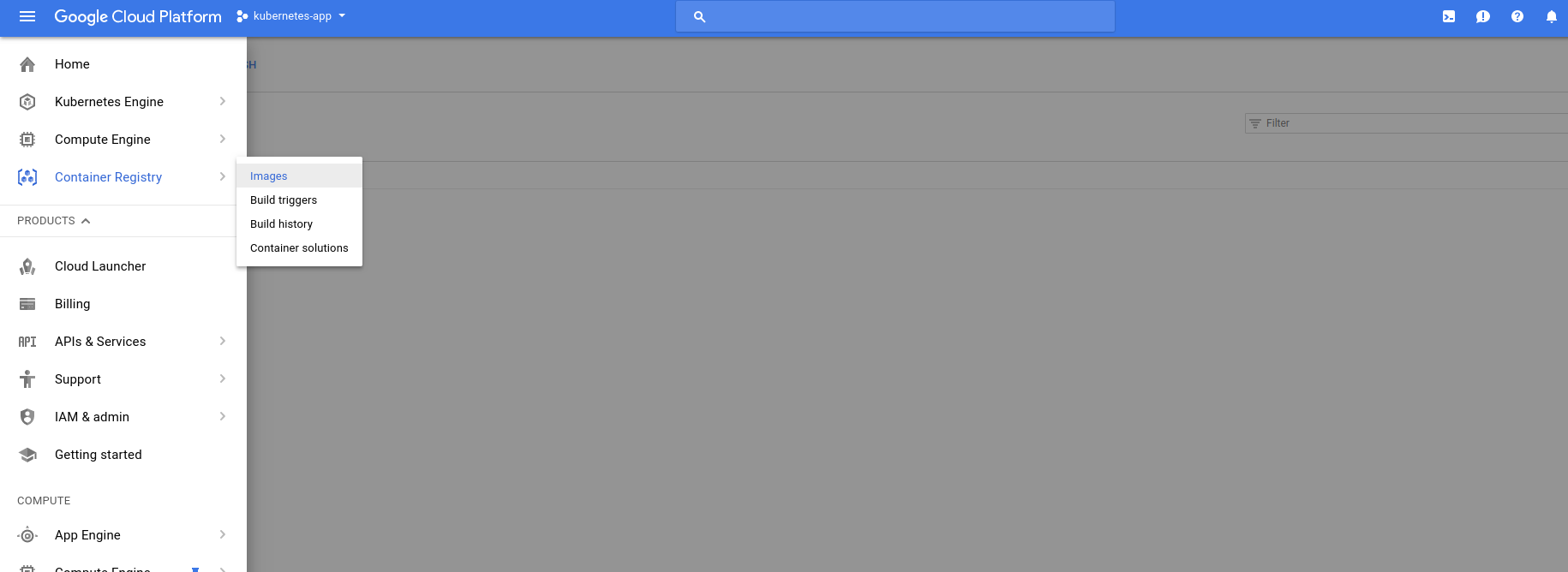
For example:

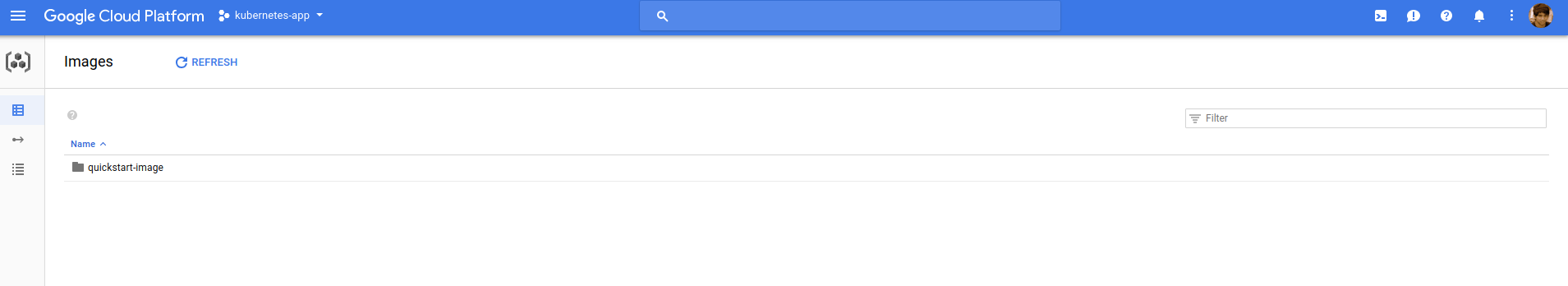
gcloud docker -- push gcr.io/my-project/quickstart-image





<https://cloud.google.com/container-registry/docs/quickstart> : REf

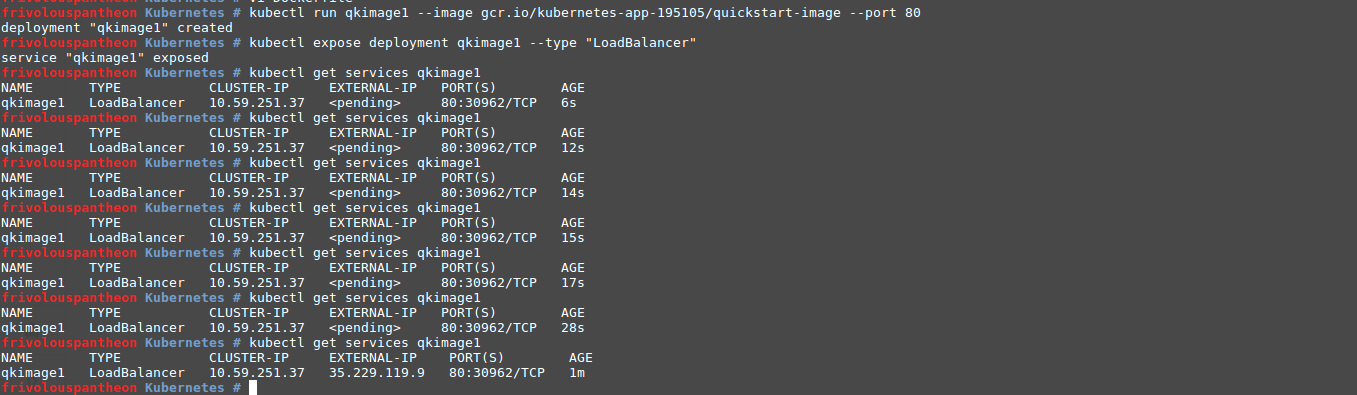




## 4. Deploying an application to the cluster

Now that you have created a cluster, you can deploy a [containerized application](https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview#workloads) to it.

Next,

The kubectl run creates a new deployment name qkimage1. The deployment’s pod runs the quickstart-images in its container.

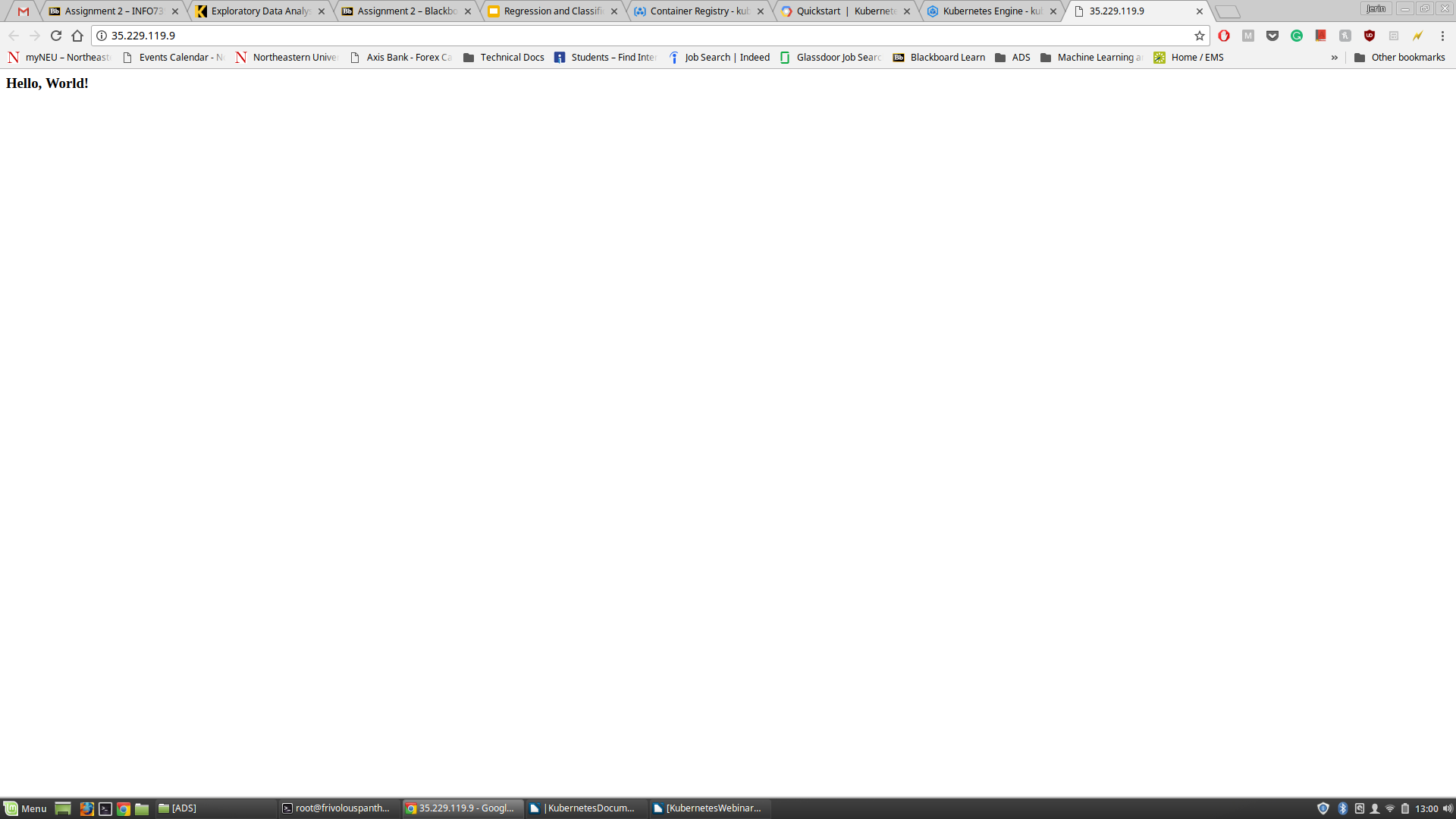
In this command:

* **--image** specifies a container image to deploy. In this case, the command pulls the example image from a [Google Container Registry](https://cloud.google.com/container-registry/docs) bucket, **gcr.io/kubernetes-app-195105/quickstart-image** indicates the specific image version to pull. If a version is not specified, the latest version is used.
* **--port** specifies the port that the container exposes.

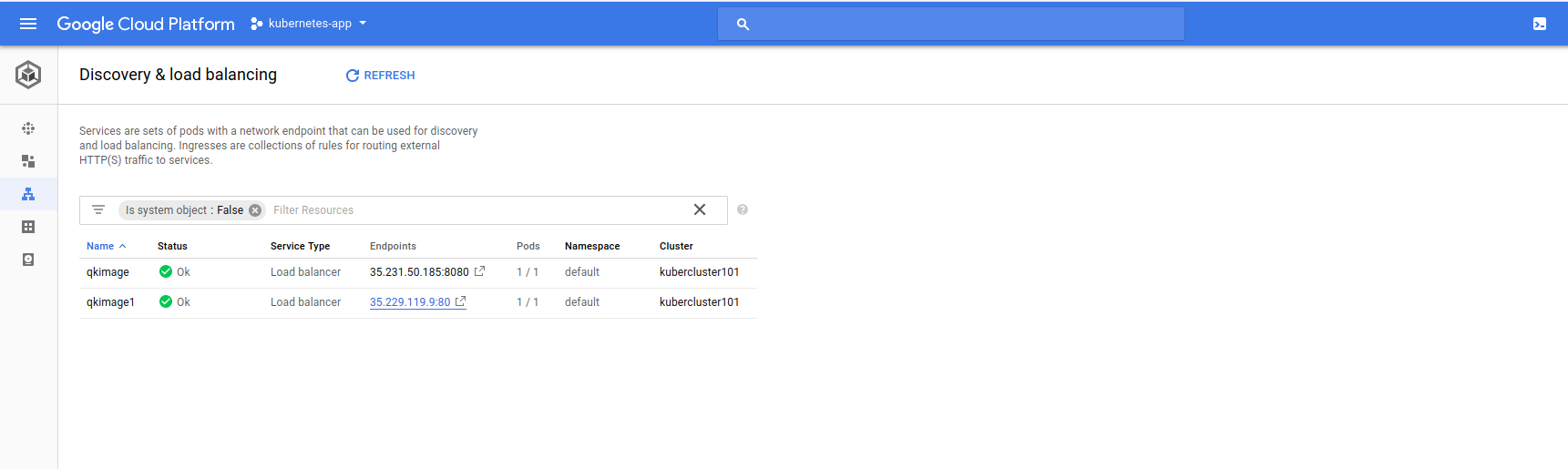
### Exposing the Deployment

After deploying the application, you need to expose it to the Internet so that users can access it. You can expose your application by creating a Service, a Kubernetes resource that exposes your application to external traffic.

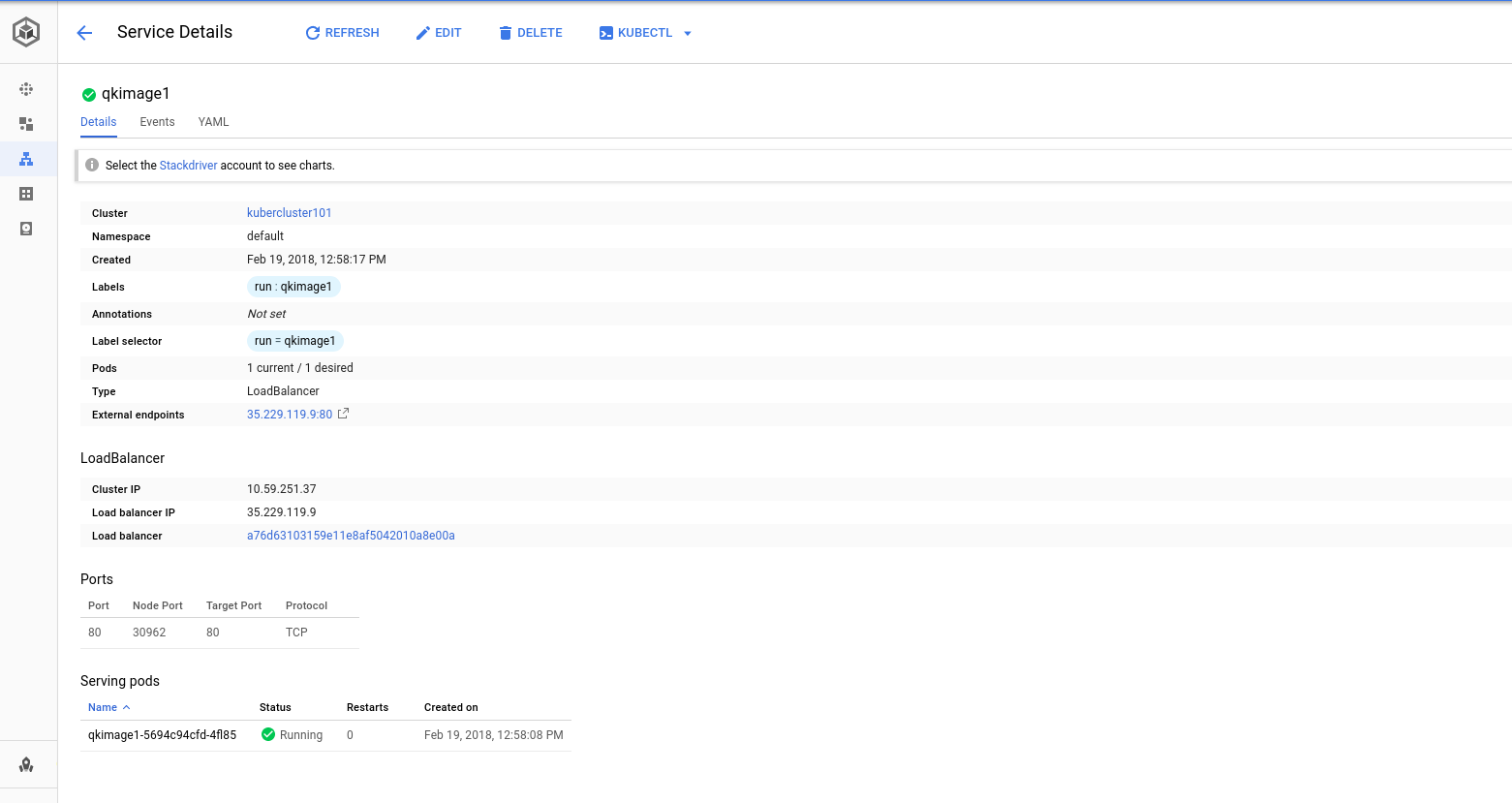
Passing in the **--type "LoadBalancer"** flag creates a Compute Engine load balancer for your container.



For Load Balancing :

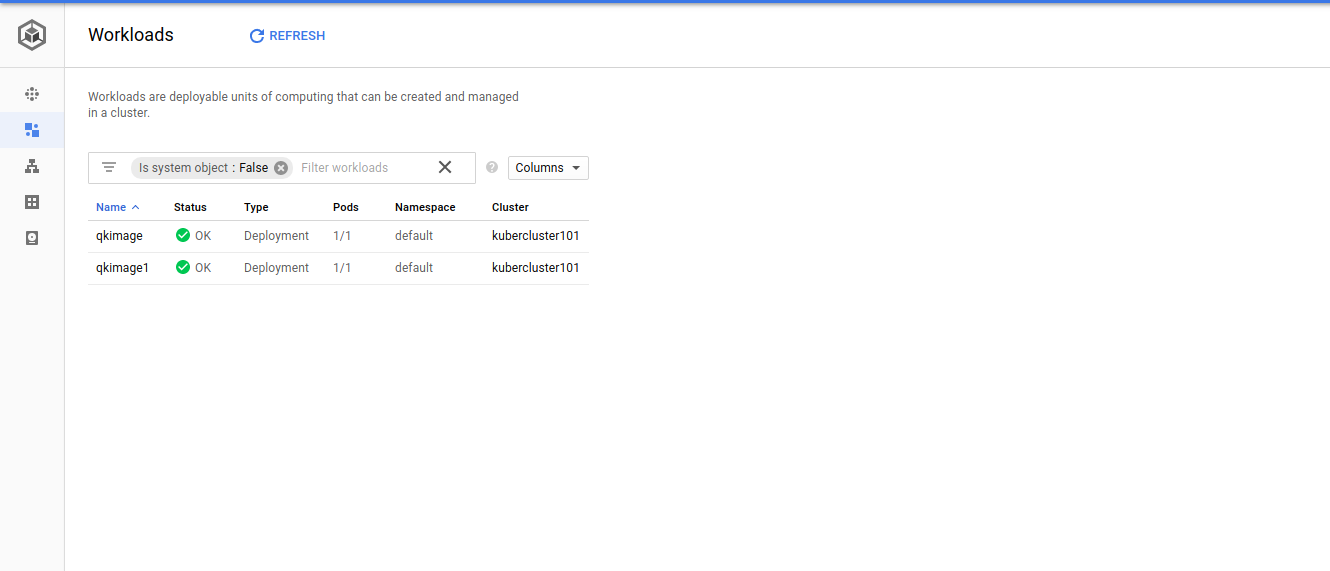


Right now its using only 1 pod to run the service

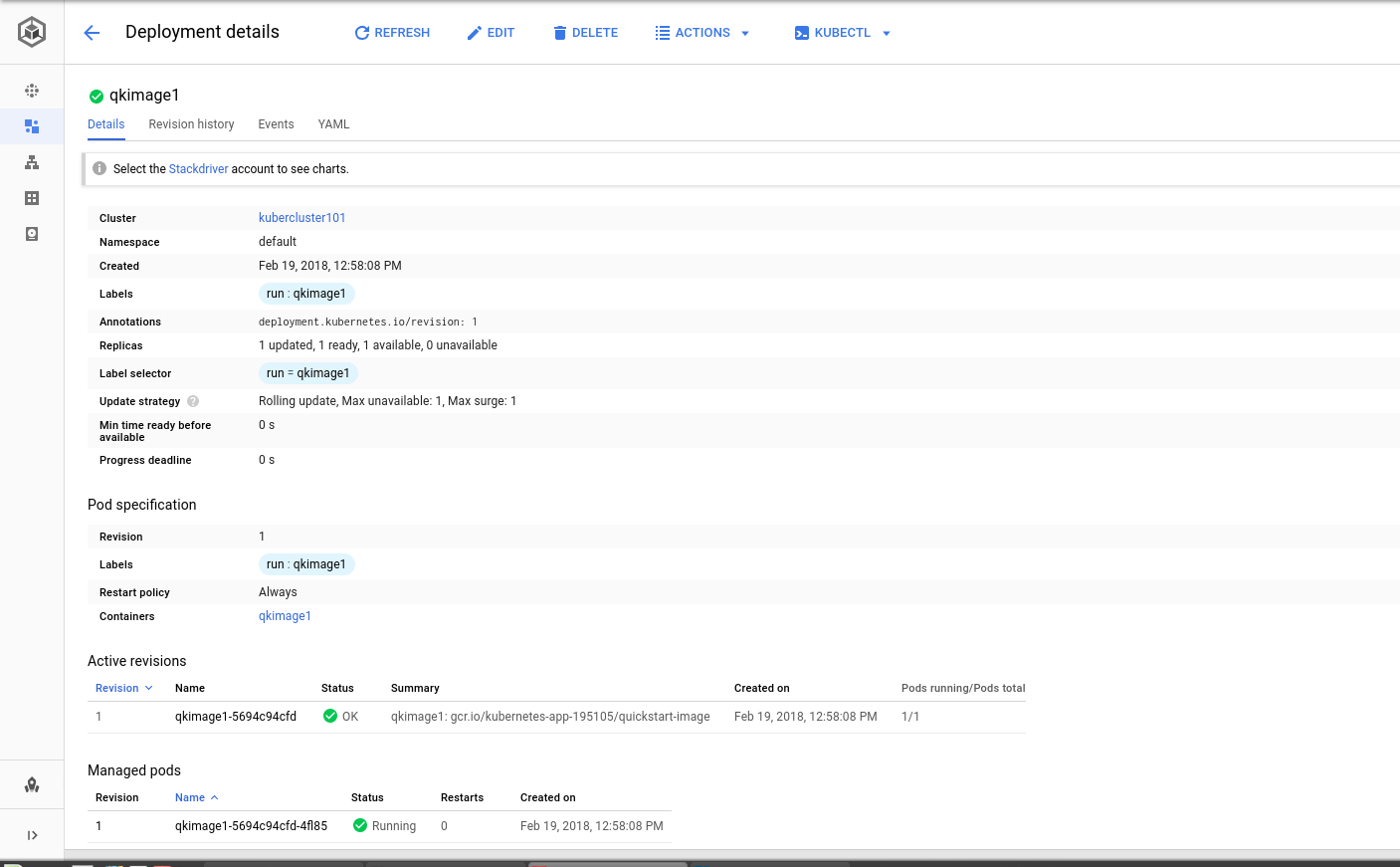


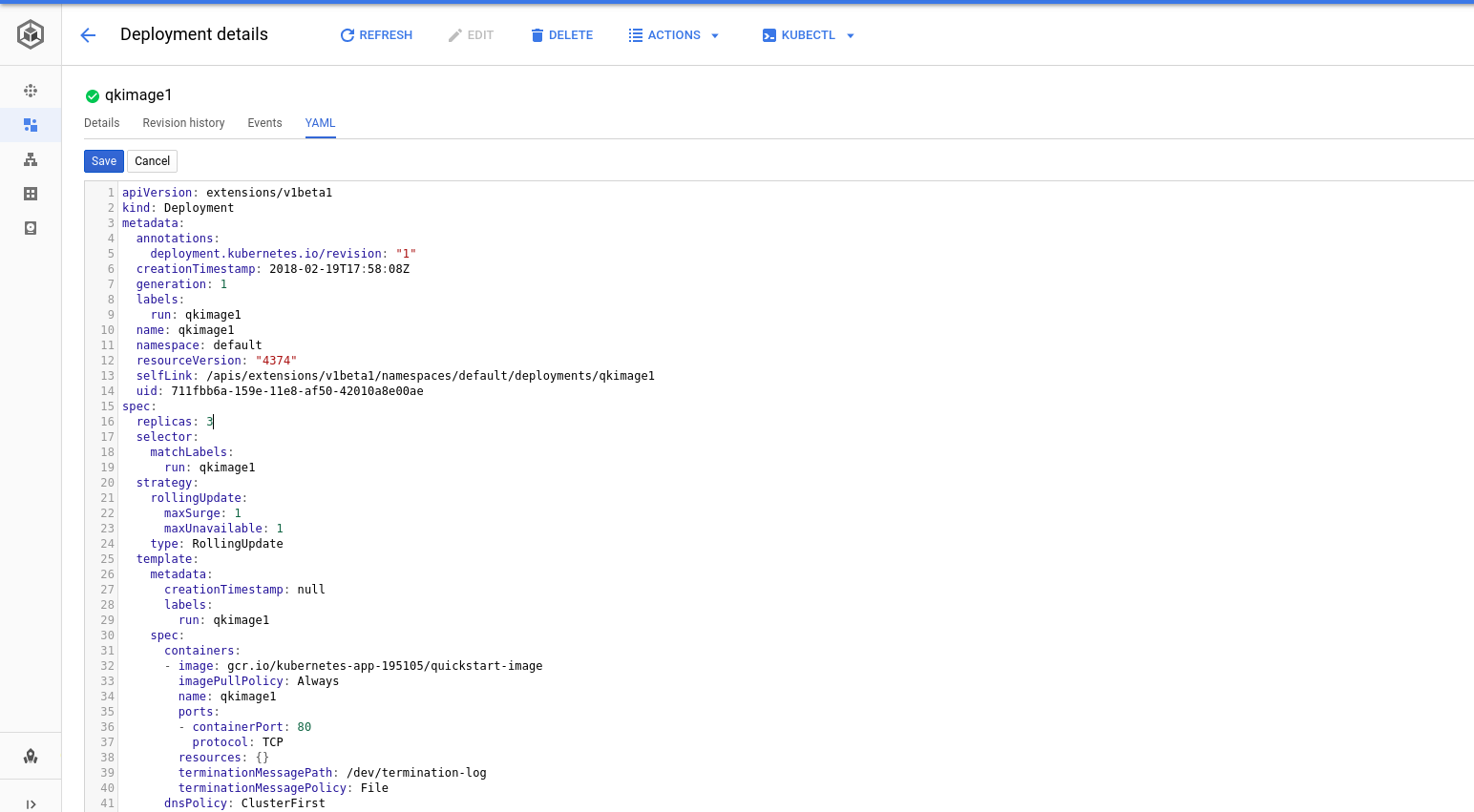
To increase serving pods

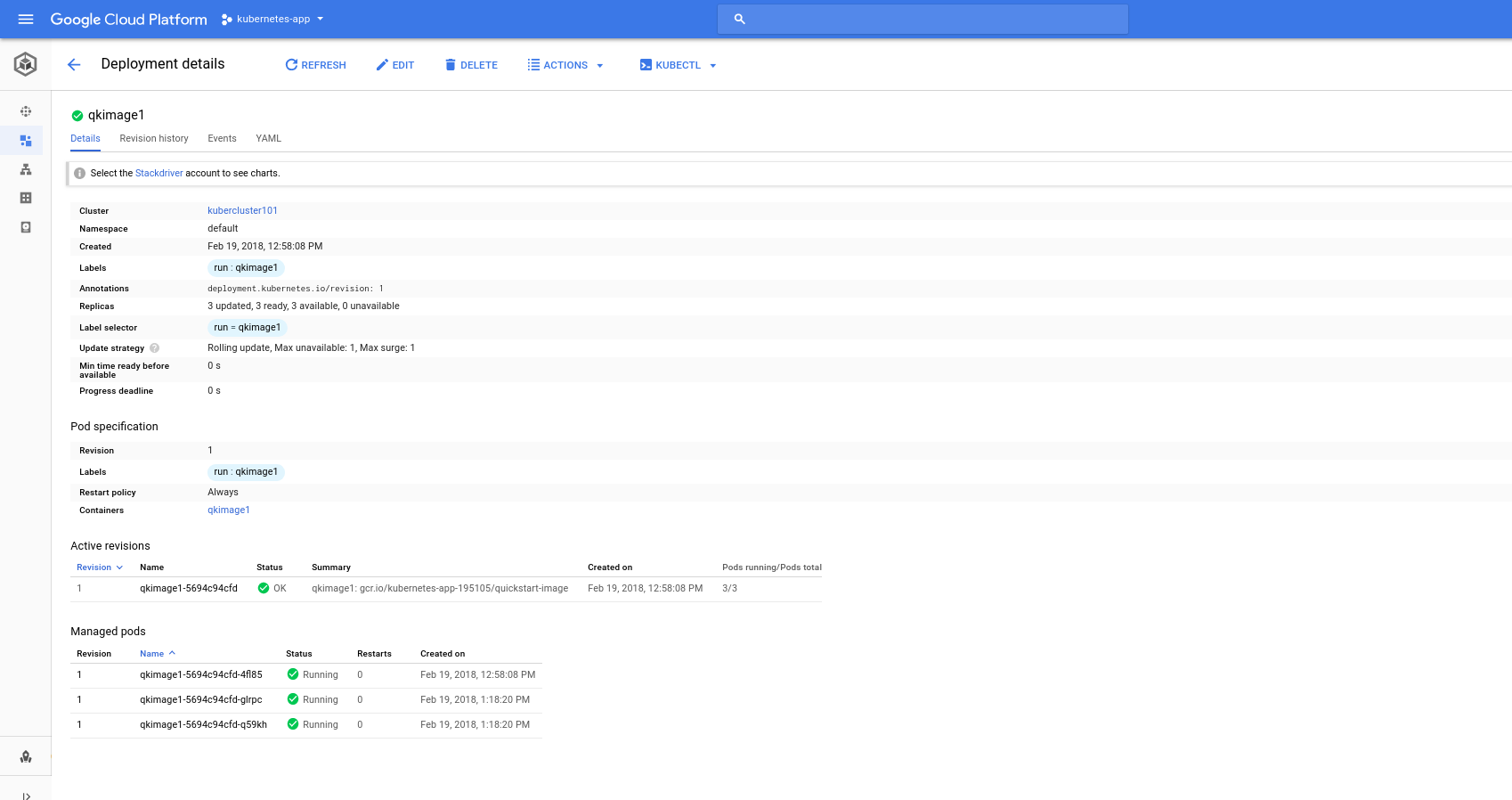
Goto kubernetes Engine> Workloads > Click the qkimage1 > edit

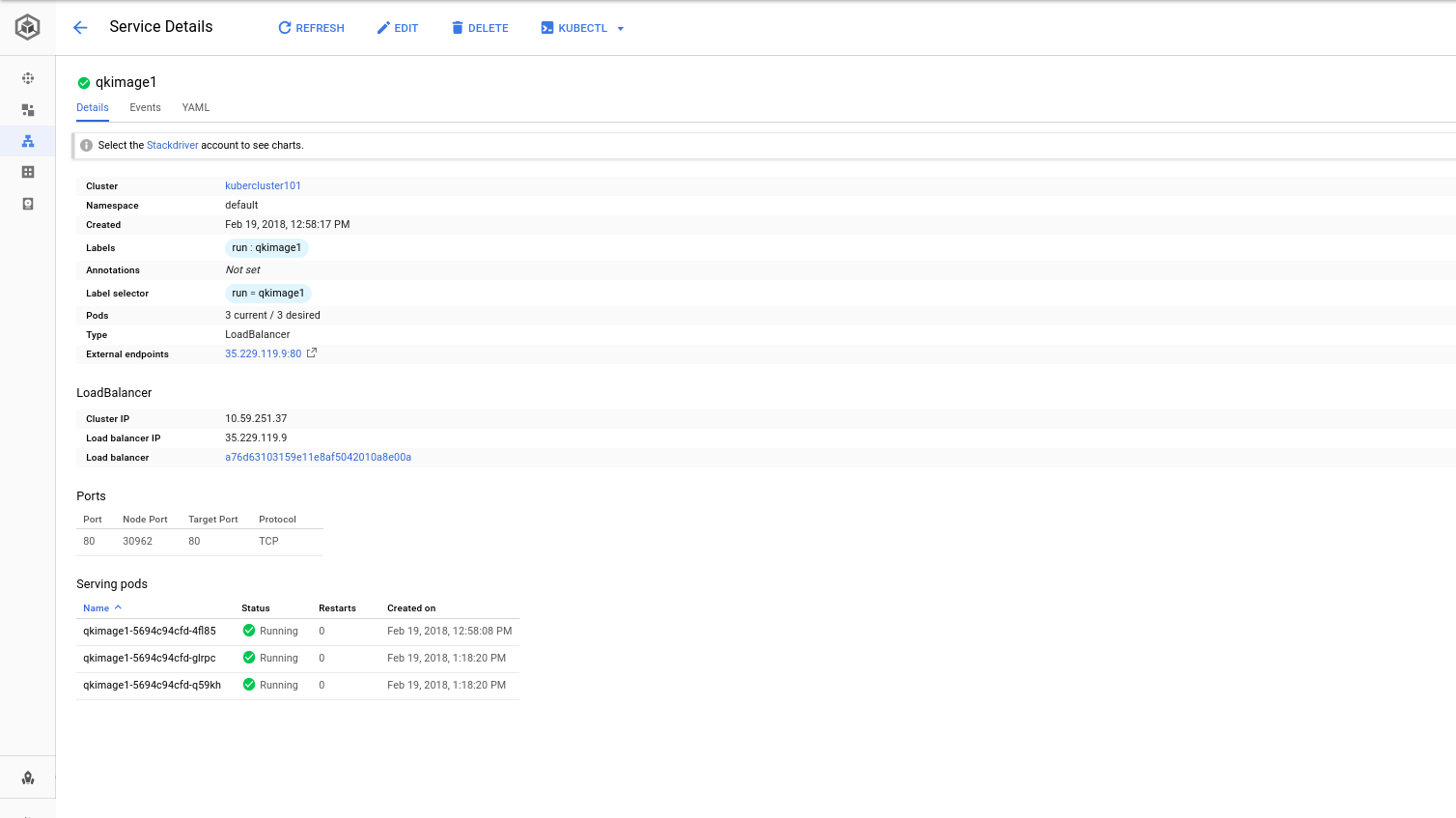


-







Now the service will be available in all the three pods which will be running the service qkimage1 and image quickstart-image.